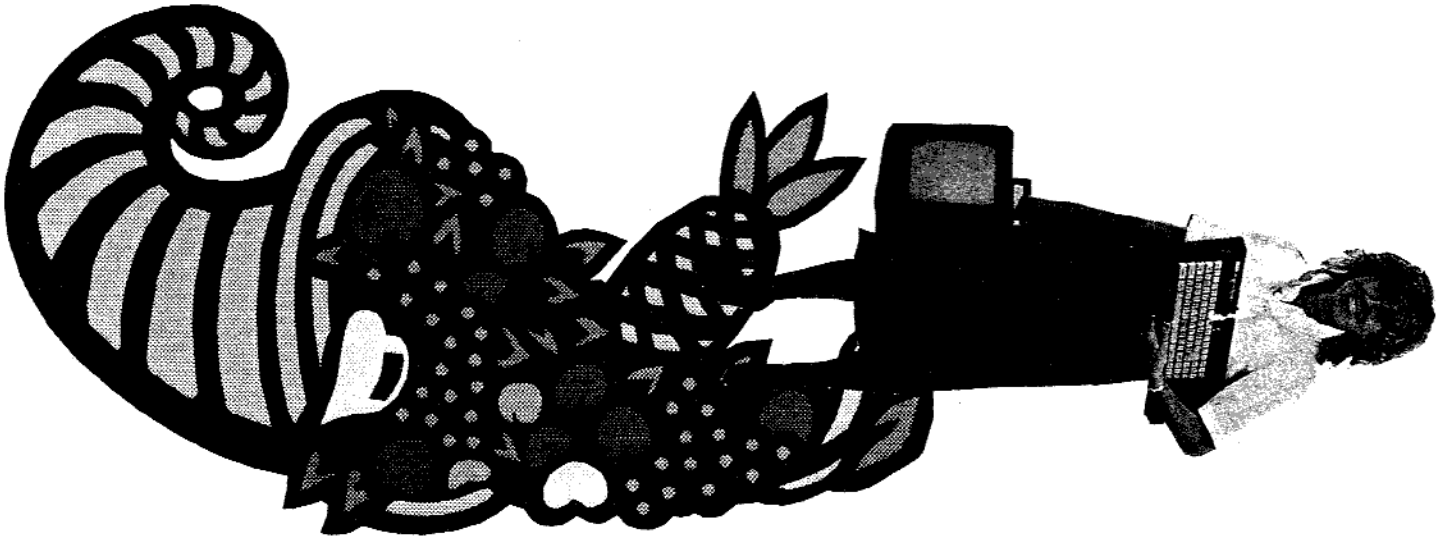
ADVENTURE CODER

Issue 24 December 1995



CONTRIBUTIONS

These are always welcome, and keep the magazine going! If you can, PRINT your text on white A4 paper in dark ink, with a margin of about an inch around each edge. Handwritten text is also welcome! As is text from a computer, saved onto a 3½" floppy disk, in the ASCII format. All disks returned.

ABOUT THIS MAGAZINE...

Adventure Coder is an amateur, non-profit publication appearing on an irregular basis. Items credited to an author (and not the Editor) remain the copyright of that author, who is free to use the same item elsewhere. All other items\pages © C.Hester 1995.

PLEASE NOTE...

The views expressed within this magazine may not correspond to those of the Editor, C.Hester.

LETTERS

If you don't want your letter to be printed, put "Not for publication" on the top. If you would like a personal reply, please enclose a stamped addressed envelope. Line ads are also accepted FREE. However, the Editor retains the right to refuse to print line ads liable to be potentially damaging, illegal, or in bad taste.

CODER

Issue 24 December 1995

SOUTH SUBURBAN, IL 604
DM
15 OCT
1991
By air mail
Par avion
WORLDPOST
UNITED STATES POSTAL SERVICE
PAR AVION
AIR MAIL
LABEL 19-A, JUNE 1980
Mark X here if a gift
Il s'agit d'un cadeau
or a sample of merchandise
d'un échantillon de marchandises
Value:
Valeur
Weight:
Poids
PS Form 2976, Feb. 1989
ROYAL MAIL
POSTAGE PAID
POSTAGE
PAID

PRINTED MATTER

Careers Service

FIRST CLASS

LIVERPOOL

Is a mechanised

Letter Offic

PLEASE

USE POSTCO

1ST

DELIVERED BY

HAND TO

SAVE MONEY

...TY ATTEMPTED	27/6
HELD PENDING INSTRUCTIONS	
RE-DELIVERY ATTEMPTED	

RETURN TO SENDER - DUE TO :-

GONE AWAY / HOUSE EMPTY	☐
REFUSED	☐
INSUFFICIENT / INCORRECT ADDRESS	☐
NOT CALLED FOR	☐

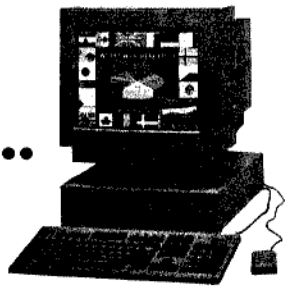
NAME AW DUTY/ROUTE No:

"Man did not weave the web of life -
he is simply one strand in it."

"Whatever he does to the web,
he does to himself."

"This we know:

all things are
connected."



Red Indian
Chief Seattle,
1854

We investigate the Internet... p10

Editorial

ADVENTURE CODER ISSUE 24

Welcome to the magazine! Whether you're an expert with programming, or a beginner starting out, I hope you find something of use here.

Firstly there's a large article about the new phenomenon called the Internet - if you've ever wondered what that's all about, now's your chance to find out.

There's also a review of the CAG adventure writing program, which is shareware, and has also been featured on a cover disk.

AMOS users can read the latest installment of their column, while fans of Stephen Groves' on-going adventure writing program can check out the eighth part of his series.

For a change from programming and technical things, I've included a short story on one page that you might like.

This issue is to be my last as Editor. I realised there was not enough material to carry on, and the magazine would have had to fold, were it not for the timely intervention of a new Editor, in the form of Alec Carswell! He has kindly agreed to take over the magazine and give it that burst of extra life that it needs. I have no doubt that he will do a great job. You might even find it starts to appear regularly again!

The magazine is to be relaunched under the new name ***Adventure Coder II - The Preceptor.*** *Alec says that "regular readers will be pleased to know that most of the old favourites will still be there... as well as some new sections."*

So, the new address will be:

16 Montgomery Avenue

Beith

Ayrshire

KA15 1EL

All that remains then for me to say is..... so long, and thanks for all the articles!

Chris

LETTERS

I'm writing to say how much I enjoy the magazine. It's kept going for longer than many fanzines and similar magazines that I know, which have long since bitten the dust. It's a shame that Coder isn't more frequent - can't we have an issue, say, every two months or something? Having said that, at least it does arrive in the end!

What are your plans for the future? Will Coder keep on going, or will it run out of steam one day (let's hope not!)? Sorry if I'm asking a question you might not know the answer yourself to!

All the best then, and keep up the good work.

Kevin Painter, Dorset.

Ed - As for my future plans for Adventure Coder, I think you'd better read the Editorial! I too hope Coder - in whatever form - carries on.

I want to say how I'm getting fed up waiting for each issue of your magazine. Surely it can't take that long to produce? If you haven't got enough articles from the readers, can't you write some yourself, to fill in the gaps!? Otherwise, we (the readers) are sat here waiting forever until the next issue is out - how many issues is that a year now? Two? If I recall, it used to be monthly!!! I don't wish to be too critical, but I'm sure you see my point.

By the way, besides the wait, the magazine is generally of a very good standard, although some of the articles tend to be a bit long.

Mark Southcliff, Hull.

Ed - When you realise that I am also working full-time, often late into the evening, then you'll perhaps see why it is hard to find enough time to produce each issue. As you say, I don't have enough articles from the readers. Each issue I do indeed have to write some things myself to (as you put it) "fill in the gaps", but clearly that takes a lot more time. In the past, there were so many articles available that I could fill an issue quite quickly, and be on the way to filling the next. Not these days!

Again, I would say to read the Editorial in this issue, which should cheer you up!

The Internet: Pardon? What's that then?

By Christopher Hester

The quotation "This we know: all things are connected" dates back to 1854, yet in 1995 it seems more apt than ever. You see, there's this big world-wide group of computer systems that have all joined together, and become known as the 'Internet'. But it isn't a fixed shape or size - new people are adding to it all the time from right the way around the globe. What's more, you can access any part of it, no matter where you are. It's the world's first truly international, borderless communications system.

How does it all work then?

It's simple. Computers with large enough storage space have been made available to the outside world by linking them to everyday telephone lines. Initially, it all started as a project in America to connect military computers during the Cold War, in such a way that one computer could transmit vital information to another computer, in the event of impending attack. This way, the Americans could keep going, by constantly moving their secret HQ of information between various computers across the land. Eventually it proved to be so useful for passing general information between distant points, that it soon spread to include non-military computers as well. After that, it was only a matter of time before the rest of the world joined in, and a standard was created to allow everyone's computer systems to cooperate. The emphasis today has shifted completely away from the military, to make the Internet a very public and open place.

Can anyone join in?

This is the good part: absolutely anyone is able to access the Internet. It is open to all, no matter what race, sex, age... so long as you can just about use a keyboard to type with, then you're in. If you work as a labourer, you've the same right to use the Internet as a millionaire executive. At the moment, it's estimated that between 20 and 30 million people are using the Internet, right across the world. So you'll never feel alone.

What's the point, though?

Why are all these people bothering to use the Internet at all? What does it offer that's so special? I'll tell you. There are two basic uses for the Internet: firstly, it offers a simple means of communicating between people, no matter how far apart they live. Just like the postage service, but much, much quicker, and cheaper! If I want to send a letter to a friend in New Zealand, say, it'll cost me the price of the airmail postage. The letter will also take a week or two to arrive - that's if it isn't lost, or damaged. With the Internet, you use what is known as 'e-mail', short for electronic-mail. Whatever you type up on your computer screen can be sent as e-mail, whether it's a short note, or a 20-page document. You pass your e-mail directly down the phone using your computer and a modem, to the nearest computer system in your area. From there, it is passed on to the destination, and can arrive in hours, if not minutes! It's certainly a lot faster than sending the same material by airmail. Plus you don't have to make a trek to the Post Office to get your mail weighed and priced up.

Of course, you can also receive e-mail yourself, from anyone, anywhere around the world. Not just everyday people are connected this way either. Star names and leading politicians have joined in, so you can actually send them a letter! Names such as Bill Clinton, Tony Blair, plus literally dozens of the latest music bands, comedians, actors and actresses, you name it.

The second major use of the Internet is to receive files directly into your computer. These can be complete programs, picture files, even sound files. You can also contact a huge list of areas dedicated to particular subjects. Whatever your interest, be it ballet, ice hockey, wildlife, TV programmes, music, politics, gardening, science, whatever, you will find something for you in these areas. Many are set up specifically to offer help on chosen subjects, so you can ask a question and receive a reply, if anyone has the answer. There are also areas set aside just for debate on an issue, one which may be the burning issue of that week's news. You can join in directly and add your opinion to the debate.

Is that it?

Not at all! Part of the Internet is known as the 'World Wide Web'. Someone had the bright idea of jazzing up the plain text on a blank background that you see normally on your computer screen. While this form of information is easy to read, it's hardly futuristic. Well, in the World Wide Web, or 'the Web' for short, you still see plenty of text to read, but it appears more like a printed page in a magazine might look, complete with pictures! What's more, the Web makes use of a feature known as *hypertext*. This means that key words in the text can be chosen by you to reveal more text about the key word. For instance, you might read "Bananas are popular in Spain". From there you could choose to read more about "Bananas", which might list a range of related topics to read, or you could choose "Spain", which might lead you to a range of topics about that country, even a collection of satellite photos of the land itself, or text about its culture, or the language, and so on. From there, you may see more key words that you can then use to link to more text, thus you can read up about exactly the right information that you want. It's just like a gigantic encyclopaedia, only one that contains more text and pictures than you could ever wish for! You can also choose to save any useful text you come across onto your computer, to read it again later.

Any section on the World Wide Web is known as a 'site', and there are new sites appearing every day. The better sites aren't just a few pages of dull text with the odd picture thrown in, they're a true visual experience, involving interactive images. Again, like e-mail, the top names are getting involved, including a growing number of businesses. Even some banks now have a site on the Web.

But what will it cost me?

You will need a computer, access to a telephone line, and a modem to convert your computer's data into a form that can be sent down the telephone line. Then you'll merely need an account with a company who will allow you to access the Internet. Yep, I'm afraid you do have to pay for the privilege, but it's getting cheaper all the time, and an average charge thesedays is only about £10 a month! It's the computer equipment that's the main cost, plus of course you will have to pay

for your telephone bill. That can be as cheap as 1p a minute, if you use British Telecom's Weekend Rate. The good news is that if you dial into the Internet via a local computer system, you will only ever pay the local phone rate for the phone call, no matter where you send e-mail to, or get files from. In other words, if I send an e-mail to somewhere abroad, it'll still only cost me the local rate charge! And if I connect to a computer that's on the other side of the globe, it's the same story, I'll only pay at the local rate! If I were to dial these places directly with my telephone, I would incur hefty international rate charges - not with the Internet!

If you can't afford a computer, then there are a growing number of places where you can go to use a computer linked up to the Internet, paying merely for the time you use.

What about the future of it all?

Assuming the computers can cope with the massive increase in demand, and there's no real reason why they can't, then the Internet will continue to expand at an alarming rate. Whole families are getting 'on-line' (connected) and one day it might be the case where you get frowned on if you're not on-line! Businesses will certainly all need to be on-line to trade fully in the future. Yet there is more to it than that. A growing array of interesting new uses for the Internet keep appearing. One such idea might seem silly at first, but you can now use it to transmit speech directly using a micro-phone, resulting in a two-way system emulating a traditional telephone! What on earth for, you say - you could just use the telephone as it is! But remember you are only paying for a local rate charge on your telephone bill. Do you get it yet? Well just think if you then dial an international number thousands of miles away - you're still paying at the local rate! In other words, it's a way to avoid long-distance charges altogether, and has certainly got the major telephone companies concerned! What's more, the dialogue can be encoded in such a way that it cannot be tapped into. No more bugged phone calls! However, the system is still in its infancy and may never be quite as good as a normal direct telephone call.

Another future use of the Internet is to send live video data back and forth. You can watch a live concert from your computer. Conferences

are now being held, using live video to eliminate the need for everyone to be present at the actual event. The Labour Party have plans to make all schools on-line - in the future, pupils might not actually visit a school building at all - they might just switch on their computers for the latest lesson. The thing to remember is that this is a two-way system - pupils could easily send messages, even live video of themselves in the room, back to the school, so they are still responding and playing a part in the learning process. Or it might be that schools will simply make use of the encyclopaedic resources of the World Wide Web, as a vast teaching tool.

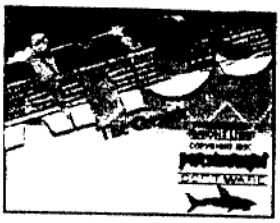
There's also talk about sending whole pieces of music to your house this way - so you may never need to visit a record store again. (You could use a printer to print out the details, even the cover sleeve. The music itself can then be played back with your computer just like a standard record.)

Then there's Internet shopping! Browse through the latest list of products available, and have them sent directly to your home. Or there's movies direct via the Internet. And a whole lot more, that we've probably yet to see.

The result of all this is one of bringing people together - whole nations in fact. The 'global village' idea of the past now seems closer than ever to reality. Of course this has political ramifications as well. People are becoming more in control of events, and more informed as well, with on-line news as it happens now available over the Internet. There are even a few daily UK newspapers offering on-line news - a bit like Teletext, only with actual pictures of the news stories as well. The Internet is breaking down the walls of the past. It's a fast approaching train, roaring over the land, through all countries of the Earth. Better get on board.....

AMOS Avenue

By John Ferris



We are about a third of the way through the adventure game listing and we are just getting to the meaty bits. What follows is the "condition" subroutine which deals with the player's instructions which have been decoded into the four variables V, AD, N(1), N(2) (verb,adverb, nouns 1 and 2).

```
CONDITION:
Rem  ---sort out synonyms now!-----
If V=0 Then Print "Sorry, I don't recognise that verb!" : Return
Rem ***do low priority now i.e. gosub low***
LP=0 : Gosub "R"+Str$(R)-" " : If LP=1 Then Return
Goto V+0
Return
```

RIGHT, first the Rem should be replaced with whatever method (if any) you are going to use to sort out the synonyms. This could be a long list along the lines of "If V=50 or V=51 then V=1" in a subroutine. Alternatively a 2 dimensional numerical array might be used, for example SY(50,1). Imagine you had 30 original verbs, these being the first 30 in the list of verbs. Imagine you had a further 20 which were synonyms of some of the first 30. The numerical array would be arranged like this: SY(verbnumber, originalverbnumber). The first thirty would be (1,1) (2,2) up to (30,30). Then if verb 31 was a synonym of verb 1, like "take" for "get", the array would read (31,1). Synonyms would be dealt with as follows:

```
If V>.30 Then V=SY(v,1)
```

Simple, eh?

Continuing with the example, CONDITION: ends if V=0, an unknown verb. The next few lines should be a nightmare to all you structured programmers out there, yes a COMPUTED GOSUB! Now when I found this to work I was mightily impressed with AMOS. Now I wanted to jump to a set of low priority routines which would handle actions that altered the status of locations. For example, in location 1, the "GET ROCK" command could caused the secret door to swing closed if the rock was wedging it open. The standard GET routine wouldn't handle this, so a low priority condition is needed. I originally had in mind an On R gosub R1,R2,R3. R1 being the conditions for Location 1, R2 for location 2 and so on. However, the STR$(R) command turns a number into a string, so the "R"+Str$(R)-" " actually computes a label which the Gosub could understand. LP is the flag which is set to 1 if a low priority condition is carried out. If it's still at 0, the program computes a goto based on the verb number. For some reason a Goto V won't work, but a V+0 will.

```
1 Rem no get all command GET
If N(1)>7 Then Print "I can't get that." : Return
HERE[N(1)] : If Param Then OC(N(1),1)=999 : Print K$ : Return
Print "I cannot get that." : Return
```

Okay, this is the GET routine which first checks if the noun involved is greater than 7, and finishes if it is, since there are only 7 GETable objects. The program checks if the object is in the same location with the HERE procedure and sets the object condition array to 999 (or carried). Prints "Okay" and finishes.

```
Rem ********* DROP*********
2 If N(1)>7 Then Print "I can't drop that." : Return
CAR[N(1)] : If Param Then OC(N(1),1)=R : Print K$ : Return
Print "I can't drop that." : Return
```

Drop is the reverse of Get (wow, such logic!). If the object is being CARried then the object condition array is set to the current location.

```
Rem ——— examine ——
3 If N(1)>7 Then Print "I find nothing special." : Return
AVAIL[N(1)] : If Param Then WRAP[EX$(N(1))] : Return
Print "I don't seem to be able to tell you anything about that." : Return
```

Examine caters for all of the GETable objects, checking if the object is AVAILable (in the same location or is being carried) before describing.

```
Rem — LOOK ——
4 Gosub ROOMPRINT : Return
```

Self explanatory really.

```
5 Rem ---INVENTORY-----
II$="I'm Carrying " : I=0
For T=1 To 7 : CAR[T] : If Param Then II$=II$+OD$(T)+"," : I=1
Next
If I=0 Then Print "Nothing at all." : Return Else Right$(II$,1)="." : WRAP[II$]
Return
```

Sets the Inventory string (II$) and the I counter. Loops through the 7 GETable objects to check if they are being carried. For each one that is, its object description plus a comma is added to II$ and I is set to 1. However, if the player is materially challenged then the routine says so and returns. Otherwise the last character of II$ is changed from a comma to a full stop. The string is printed out via the WRAP procedure and the routine ends.

Here follows the movement routines, each being very similar:

```
6 If EX(R,1)<>0 Then R=EX(R,1) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem NORTH
7 If EX(R,2)<>0 Then R=EX(R,2) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem SOUTH
8 If EX(R,3)<>0 Then R=EX(R,3) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem EAST
9 If EX(R,4)<>0 Then R=EX(R,4) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem WEST
10 If EX(R,5)<>0 Then R=EX(R,5) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem UP
11 If EX(R,6)<>0 Then R=EX(R,6) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem DOWN
12 If EX(R,7)<>0 Then R=EX(R,7) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem NE
13 If EX(R,8)<>0 Then R=EX(R,8) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem SE
14 If EX(R,9)<>0 Then R=EX(R,9) : Gosub ROOMPRINT : Return
```

```
Print M$(6) : Return : Rem NW
15 If EX(R,10)<>0 Then R=EX(R,10) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem SW
16 If EX(R,11)<>0 Then R=EX(R,11) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem IN/ENTER
17 If EX(R,12)<>0 Then R=EX(R,12) : Gosub ROOMPRINT : Return
Print M$(6) : Return : Rem OUT
```

Basically, each checks if the relevant exit in the present room is available (not 0) and if so sets the location variable R to what is found in the EXits array. ROOMPRINT does what is says and the routine finishes, the player now being in a different location. Otherwise, message 6 (M$(6)) complains "I can't move in that direction".

Talking about messages, I'll end this article with the 37 messages used in the game, I expect more to be added in the near future:

1. The staff fits easily into the hole in the floor.
2. The sunlight shining through the opening in the roof is focused by the crystal onto the east wall. There is a rumble and a section of the wall opens inwards.
3. There is an open doorway in the east wall.
4. The small rock wedges the door open.
5. I put the staff into the hole in the floor and stand back. Nothing happens. I wait. Nothing happens again. I think something should have happened. Perhaps I should have been quicker?
6. Sorry, I cannot go in that direction.
7. The light dims as the sun is no longer shining down through the hole in the roof.
8. As I pick up the staff the door closes.
9. I get the rock and the door slides shut.
10. I say 'vulco' and the crystal on the staff begins to glow with a clear white light.
11. Nothing happens.
12. The door is locked.
13. The door is open.
14. The door is unlocked but shut.
15. I catch the water in the jug. After what seems a very long time the jug fills with water.
16. The jug is already full of water.
17. I give the door a hefty kick and to my surprise it collapses. It was obviously very rotten. I can move north now.
18. I kick the smashed door but receive very little gratification.
19. The door is locked. It's also in very poor condition, the hinges are rusty, the wood 20. itself is crumbling. Tempting, isn't it?
20. The door has been smashed open and lies in several wormy pieces on the floor.
21. There is a door in the north wall.
22. A doorway leads north from here.
23. The packet is already open.
24. The packet is already closed.
25. A seed rolls onto the palm of my hand.
26. I carefully put the seed back into the packet.
27. I water the sees and stand back as magic can be dangerous, especially old stock like this. A thin silvery green shoot sprouts from the hole and two leaves spread to receive the light. Suddenly, without warning, the plant begins to grow rapidly. It rises, stem thickening, producing many large broad leaves. As it reaches the top of the pit it flops over onto the grass. I hear it crawling along the ground.
28. I take a deep breath and climb the thick fibrous stalk, finding hand and footholds easy to come by.

29. I seem to be free of that pit! The setting sun warms me and its light refreshes my spirit. That's this game over!
30. GAME OVER. Thank you for playing.
31. Hmm, the hole is home to the base of a thick fibrous stalk of a plant.
32. There is a silvery magic seed in the hole.
33. The hole is about an inch in diameter and is too uniform to be a natural feature of the floor.
34. The walls are made of sandstone and seem pretty solid. Except of course for the eastern wall which has a whopping great open door in it.
35. All four walls are made of rough sandstone and appear solid. If there is a secret passage, it's very well hidden.
36. The hole is full of staff.
37. The water is a clear liquid which drips from a crack in the roof of the tunnel.

How to Make a Thingy (Part VIII)

Hopefully, you have now seen what the final screen for the adventure will look like. Don't worry, you can always have a text only adventure if you want, and that would allow you to have more memory for text.

The next section of code starts with MAINRT. This will be the MAINRT for the final programme, so you should now delete the section called MAINRT that we entered for testing purposes.

MAINRT is the heart of this programme, it is where all the major sub-routines are called from. In essence, the next 19 lines of code make sure that everything happens in the correct order ie. screen setup, graphics, location description, prompt and input.

The majority of these first 19 lines are either CALLS, JUMPS, or conditional JUMPS. Because I feel it is very important to understand what happens here, I have decided to alter the presentation slightly for this section of code. First of all I will list the 19 lines without comments. Each line will be numbered and at the end of the listing, I will refer to each line number and give an explanation of what it does. This will enable me to make a fuller explanation than would be possible at the end of each instruction.

```
01   MAINRT   CALL TABUP
02            CALL NEWLOC
03            CALL PROC.1
04            JP PRMPTC
05   N.LOCC   CALL NEWLOC
06   PROC1C   CALL PROC.1
07   PROC2C   CALL PROC.2
08   PRMPTC   CALL PROMPT
09   INPUTC   CALL INPUT0
10            CALL UPTURN
11   WORDC    CALL VBS2
12   RESPC    CALL RESP
13            BIT 2,(IX+14H)
14            JP NZ,WORDC
15   NOMOR2   XOR A
16            LD (IX+1EH),A
```

```
17              BIT 7,(IX+14H)

18    SMINVC    CALL NZ,SMINV

19              JP PROC2C
```

There it is then. Not a lot to it was there. Now lets see if I can make some sense of what each line does.

01 CALL TABUP

Quite a simple routine that copies the contents of MAINOB, MAINEX and MAINFL into OBJPOS, EXITS and FLAGS. As explained earlier the tables preceded by MAIN__ are the masters. The other three are the working copies that can be changed as the game progresses. Obviously this CALL is made before the game begins.

02 CALL NEWLOC

As the name implies, this is a routine that deals with new locations. The first thing it does is to check if the location is dark. Flag 0 = 1 if it is dark. If the location is dark, and there is no lamp, then no description or graphics are printed. If the location is light (Flag 0 = 0) a check is then made to see if we are in text only or graphics mode. The location number is then calculated, and the location description is printed before a RETurn is made.

03 CALL PROC.1

PROC.1 is a routine that carries out actions immediately after the location description but before the the input prompt. The data to be executed by PROC.1 is held in a user defined table called ONEDAT. As an example, if you want objects at the current location to be listed, this is carried out by PROC.1. In general, actions carried out by PROC.1 are not conditional and are only carried out after a location description.

04 JP PRMPTC

A straight forward jump to line 08 after the first location of the game has been described.

05 CALL NEWLOC

Again, a CALL to describe a new location. (See line 02)

06 CALL PROC.1

See line 03.

07 CALL PROC.2

PROC.2 is a routine that carries out actions that are held in the table TWODAT. The actions are carried out before the input prompt is printed, but after the last input has been dealt with ie. it is not used for the very first location, but is used every time after that. Actions carried out by PROC.2 can be conditional, but generally they are conditional only on the state of flags and are not conditional on input. PROC.2 would be used to show that something has happened since the last player input eg. a person you were talking to has walked away, or a bird landing in a tree.

08 CALL PROMPT

The routine that chooses one of four prompts and displays it on the screen.

09 CALL INPUT0

That is by the way INPUT(zero). This is the routine that prints what you type onto the screen, and enters it into a buffer for checking against known words. All the letters A-Z are recognised together with the delete and enter keys. Symbol shift, extend mode and graphics are all ignored.

10 CALL UPTURN

The routine that updates flags 17H and 18H which keeps count of the number of turns taken.

11 CALL VBS2

VBS2 is the start of the routine that deciphers what is in the input buffer. It is commonly known as a parser. Obviously, the parser will recognise whatever words you programme it to. A full description of the parser will be given when we get to the routine VBS2.

12 CALL RESP

RESP, is the routine that is used to make actions happen. If the input command sentence consists of OPEN DOOR, then RESP will check that there is a door at this location, that you are also at this location, that the door is closed, that you have the necessary implements to open the door AND if all these conditions are met, will show the door to be open. Again, a full description will be given when we get there.

13 BIT 2,(IX+14H)

This instruction checks bit 2 of a flag to see if there is more text in the input buffer that needs deciphering. If there is, the flag is set and a conditional jump is made back to WORDC (11). If there isn't any more text in the buffer, the jump fails and the instruction at (15) is carried out.

14 JP NZ,WORDC

The conditional jump to WORDC.

15 XOR A

A simple, single byte instruction to load A with zero. This instruction is only carried out when there are no more words in the input buffer.

16 LD (IX+15H),A

Load the zero into the flag that contains the duplicate verb.

17 BIT 7,(IX-14H)

Test the flag bit to see if graphics are being drawn.

18 CALL NZ,SMINV

If the bit is set, we are in graphics mode and we need a call to print the small inventory.

19 JP PROC2C

Return to line 07 to repeat the process again.

When the game is first initialised, the screen is set up through TABUP. The first location is then described by NEWLOC, this is followed by PROC.1 which will take care of describing items at the location etc. A jump is the made to print the prompt and input is then accepted. The number of turns taken is then incremented and then the input buffer is deciphered. If a command sentence is recognised then any responses to the input are carried out. A check is then made to see if the input buffer is empty, if it isn't, a jump is made to line 11 where he deciphering continues. This process continues until the buffer is empty, when a jump is made to line 07 and the whole process is repeated.

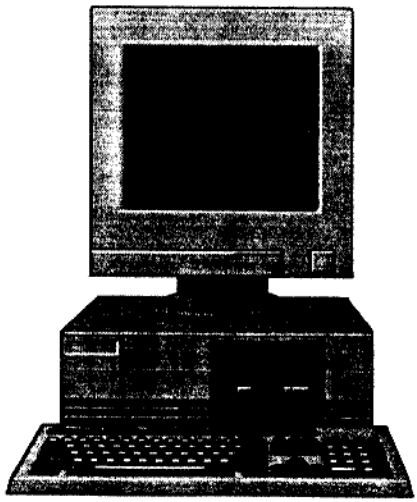
As can be seen, the whole game is controlled by these 19 lines of code. Although CALLs and sub-CALLs are made to many different routines, the programme always returns here until the game is terminated. It may seem a strange place to put the main routine of the programme, but that's the order in which it was written. I do know that it doesn't follow the rules of structured programming, but I can assure you that it all works.

We can now continue with the rest of the code. The first thing that we shall do is check if the location is light or dark, this is followed by a check to see if we are in graphics or text mode.

```
NEWLOC          BIT 0,(IX+0)     ;Entry point from MAINRT
                                  that checks flag 0 to see
                                  if the location is dark.
                JP NZ,DARK       ;Jump if the flag is set
                                  (ie location dark)
N.DARK1         BIT 7,(IX+1AH)   ;Check the duplicate
                                  graphics flag (1=draw)
                JR NZ,BITS1      ;Jump to BITS1 if graphics
                                  are to be drawn
                JR Z,BITS2       ;Jumpto BITS2 if graphics
                                  are NOT to be drawn

;We return here from both BITS1 and BITS2, and now know that
the location has light and the graphics flags are correctly
set. So..

NOTDRK          BIT 7,(IX+14H)   ;Check whether to draw the
                                  graphics
                JP NZ,PICS       ;and jump to draw them...

                CALL NOPICS      ;or make a call to set the
                                  screen co-ordinates for
                                  printing at the top of the
                                  screen

;We either return here from the CALL NOPICS above, or from
PICS. The screen is either completely blank in text mode, or
we have the border, exit arrows, small inventory and graphics
on the screen and we are ready to print the text. This
routine calculates the address in memory whre the location
text is held.

GETLOC          LD B,0           ;}By loading B with zero
                                 ;}and C with the contents
                LD C,(IX+0AH)    ;}of the current location
                                 ;}flag, we end up with the
                                 ;}current location in BC
                LD HL,LOCTB1     ;Load HL with the base
                                  address of LOCTB1
                ADD HL,BC        ;Add the two together
                                  leaving HL pointing at the
                                  low byte of the text
                                  address....
```

```
                    LD E,(HL)       ;....which is then loaded
                                     into E
                    LD BC,96H       ;The length of LOCTB1

                    ADD HL,BC       ;HL will now point at the
                                     high byte of the text
                                     address...
                    LD D,(HL)       ;...which is loaded into D

                    EX HL,DE        ;HL now contains the
                                     address of the text for
                                     this location
                    JP PLOCR        ;Jump to print the text


;BITS1 and BITS2 either set or reset the main graphics flag,
depending on whether graphics are required.

BITS1               SET 7,(IX+14H)  ;Set the main graphics flag

                    JR NOTDRK       ;Jump to find and print the
                                     location text
BITS2               RES 7,(IX+14H)  ;Reset the main graphics
                                     flag
                    JR NOTDRK       ;Jump to find and print the
                                     location text


;If the location is dark, we come here from NEWLOC. Our first
check is to see if the lamp is lit. If it isn't, we print the
message contained at DARKWD. If the lamp is lit, we have to
check is it
a) carried
b) at this location but not carried (it would still provide
light)
c) neither of the above

If it is either (a) or (b), we can return to N.DRK1 to print
the text.
If it is (c), then we continue and print the message at
DARKWD


DARK                BIT 0,(IX+01H)  ;Knowing the location is
                                     dark, we check if the lamp
                                     is lit
                    JR Z,DARK2      ;If the lamp is UNLIT, we
                                     jump to print the message
;We now know the lamp is lit, but where is it

                    LD A,(OBJPOS+1) ;The lamp is object No.1
                                     so load it's location into
                                     register A...
                    CP 0FFH         ;..and check if it is
                                     carried
                    JP Z,N.DRK1     ;Jump to describe the
                                     location if it is
                    CP (IX+0AH)     ;Compare the lamp's
                                     location with the current
                                     location
```

```
                JP Z,N.DRK1     ;Jump to describe the
                                 location if they match

;We come here if the lamp is unlit, not carried or not at the
current location. We must now clear the whole screen and
print the message "It's too dark to see anything"

DARK2           CALL NOPICS     ;We make a call to clear
                                 the screen and set the
                                 print position at 0,0
                LD HL,DARKWD    ;The address of the message

                CALL PLOCR      ;Normaly we jump to PLOCR
                                 and return direct from
                POP HL           there. As this is a CALL
                                 we have to remove the
                                 return address from the
                                 stack
                RES 7,(IX+14H)  ;Reset the graphics flag
                                 because we cannot have
                                 graphics with no light
                JP PRMPTC       ;JUMP to return to MAINRT

DARKWD          DEFM "It's too dark to see anything"

                DEFB 0DH        ;The return character to
                                 signify the end of
                                 printing
```



CAG - A Shareware Adventure Writing Utility

Reviewed by John Ferris

I found this program on the February 1993 Amiga Computing cover disk, where it received barely a mention in the magazine.

CAG (Create Adventure Games) V 1.2 was written by Marco A. G. Pinto of Portugal using compiled Amos BASIC. Included with the program are two very small demos, a default file and reasonable documentation. Reading the documentation was interesting as Mr Pinto's English is good, despite some novel spelling and sentence construction. My favorite was the use of "bellow" instead of "below" which would not have been so bad except CAG has a command by that name. Don't get me wrong, I am not mocking Mr Pinto's language skills as my knowledge of his language is almost nil, credit where credit's due and all that.

CAG is somewhat similar to GAC/PAW/Quill in the way it works, tables of conditions are built up to deal with the vocabulary, for example:

GET AXE: HERE 1 GET 1 OK END

In other words, if "GET AXE" is typed and object 1 is present then it is picked up. However, there does not seem to be any way of automating this so if you have fifty objects they must all have individual GET, DROP and EXAMINE conditions. Seems like hard work to me.

The commands available look quite comprehensive, I noticed a number of commands straight out of GAC, including those which dealt with the counters. The program is also able to incorporate graphics into the game, loaded either from disk or contained in the adventure database itself. It also allows the mouse to be used in a limited fashion in the game. One of the demo files has you opening and closing a graphic door with the mouse, OK for effect but not a great enhancement to a game. Music is also catered for, Soundtracker modules (music files from a PD music program) can be played as background music.

Room descriptions are entered using a simple but effective text editor, exits are entered as "VERB(destination)" on a special line below the description, reminding me of Hatrack by Heyley Software. Vocabulary is defined as verb, noun, both, adjective or adverb and is entered into a table along with a number. Synonyms of a word are given the same number.

Yes, looks promising, doesn't it? On paper it is, but in RAM it had a nasty habit of crashing with almost indecent haste when I tried to use it. It tends to do nasty things to the adventure database, rendering useless up to minutes of hard work. I said "minutes" and I mean minutes. CAG was only stable for a very short time before crashing on some action which had worked just seconds before. I even disconnected the hard drive (inc 2MEG RAM) just to see if it was anything to do with that. If anything, CAG performed worse with a record crash time.

Unfortunately what we have here is a seriously untested program which I have to say is too unstable to be of any use. This is a shame really because there are a number of examples of good programming (eg. the text editor) which I admired, but they were let down by the bugs. CAG is the sort of program which gives PD/Shareware a bad name. I really wonder how it ever got onto the cover disk of a glossy magazine? Did they test it or what? I think I know the answer and it bodes ill for that magazines reputation. I am only grateful that I didn't "buy" CAG from a PD library.

If the program was fully tested and presented bug-free I would say that it was definitely worth a look for the would-be adventure author on a tight budget. The only reservation I have is the lack of automated GET/DROP/etc commands which makes for a lot of drudgery. It isn't as powerful as Hatrack (the only COMMERCIAL rival in the adventure utility market I know of) but it still looks potentially useful. I will be interested to see how stable later versions of this program will be.

Light Armageddon

A short story by Christopher Hester

Contrary to popular belief, the President cannot 'push the button down' to initiate a nuclear attack. The 'button' is in fact two switches at separate locations. These can only be activated together, at the command of the President himself.

We crashed into the corridor and blasted the guards away. They vaporised nicely. This was it - the last of the two switches that could spell doom for Mankind. Recent events on the world stage by unstable countries had put the world on edge. Public pressure had failed to prevent the President from giving a threatening speech, clearly indicating he was prepared to strike first. This was to "preserve world peace" of course - in reality, to obliterate the enemy before they could attack. Such was the confidence in his new system to destroy incoming missiles, he was now willing to provoke anyone - like a kid with a gun, who felt he had the perfect bullet-proof vest.

It felt like the future of the whole planet could be in our hands. If we could deactivate both switches, the President would be forced to decline from the aggressive position he had taken up, and peace would have a chance. The first switch had been successfully deactivated already, but that left us in the dangerous state where it would only take the second switch to launch a nuclear attack.

The sweat poured from our brows - we had the right location, but where was the second switch? After blasting down the door infront of us, we entered a dark room. Inside, all we could see was the shape of the walls lit by the outside light. Perhaps this switch had been disguised to stop anyone like us from finding it. Perhaps... perhaps there only was one switch after all...

We pressed on through an open door at the far end of the room. This led to a room in total darkness.

It was no use. My colleague had had enough. It had been a tiring mission for us all. He wanted it over fast. Moving across the walls with his hands, he eventually found something protruding in the darkness.

"Hold on while I turn on the light." he said. "I can feel the switch on the wall here..."

BACK ISSUES

These are available at the following reduced prices:

£1.00 - Britain **£1.40** - Europe
£1.85 - Zone 1 (Canada, Egypt, Falklands, Saudi Arabia, USA, etc.)
£1.95 - Zone 2 (New Zealand, Australia, etc.)

Payment is by the usual manner - cheques payable to "C.Hester", at the usual address, or you can use a Postal Order, International Giro, or British decimal stamps for small amounts. If you want more than one back issue, then just mutliply the cost by the number of issues you want - for example, 5 issues (if you live in Egypt) is 5 times £1.85, equalling £9.25.

ISSUE 1 - Utilities & Add-Ons, GAC+ review, PAW (available exits), Z80 Machine Code (Z80 overview), Whatever Happened To... *Valley Of The Source*, GAC graphics (colour, perspective, ellipses, rectangles), Useful Addresses, etc.

ISSUE 2 - GAC (pokes, starting a game), GAC+ (pokes), PAW (parser, vocabulary), Z80 Machine Code (command input routine), Whatever Happened To... sound-only games, STAC (starting), Adventures (storyline, writing), Useful Addresses, Utilities & Add-Ons, etc.

ISSUE 3 - The A-Z Of RPG, PAW (cars, again, oops, find object, overlays), GAC (doors), STAC (tips, list), Spectrum Machine Code (parser), Useful Addresses, Utilities Available, etc.

ISSUE 4 - PAW (inventory, get all, drop all, containers, objects in mazes, taxis, exit printing, swear protection, again, oops, clocks, add-ons), Adventure Columns reviewed, GAC (inside info), Useful Addresses, Utilities Available, etc.

ISSUE 5 - PAW (characters, objects, telephones, examine, password, role-playing, clocks), ADLAN (objects), Play-By-Mail, Atmosphere, Tom Frost interview, Lastability, etc.

ISSUE 6 - PAW (flag addresses, mazes, password, inventory + objects, characters, start anywhere, lives), GAC (messages, character sets, wear, exit printing, characters, save + load, start-up, bugs), Whatever Happened To... Isaac Asimov adventures, ADLAN (messages), Adventures (endangered?), etc.

ISSUE 7 - BUMPER ISSUE!! - PAW (flag directory, money, oops, him/her, last), ADLAN (vocabulary), 6502 Machine Code (registers, hex, useful memory addresses), Sexism, Sandra Vogel interview, Patrick Walsh interview, Adventures Of The Future, Lastability, etc.

ISSUE 8 - PAW (exits, mazes, characters, overlays, fonts, flags, object weights), Adventure Writing For Beginners, ADLAN (light + dark), Spectrum Adventure Utilities, GAC (character with an inventory), *Adventures On The Spectrum* (Mike Gerrard) review, 6502 Machine Code (modes), New Words, STOS (ideas), etc.

ISSUE 9 - PAW (independent characters, chance/random, exchanging objects, externs, passwords, memory map, overlays), GAC/PAW (converting commands), Z80 Machine Code (room descriptions), STAC (thoughts), 6502 Machine Code (graphics screen, stack, useful commands), CBM64 Art Programs (+ useful routines for them), ADLAN (errors in manual), etc.

ISSUE 10 - PAW (character speech, do's + dont's, externs), Z80 Machine Code (exit printing), New Words, 6502 Machine Code (interrupts, handy BASIC + machine code routines), 1541 Disk Drive (secrets, sorting files, Toolkit IV secrets), ADLAN (6-room game!), STOS (adventure creator project + routines), etc.

ISSUE 11 - STAC (starting a game), PAW (input editor, characters, 128K into 48K conversion), The Ultimate Adventure Creator, *Gems* compilation review, ADVSYS review, 6502 Machine Code (routines including sprites in the border!), ADLAN (explanation of last issue's game), TALESPIN (thoughts), Adventure Languages, etc.

ISSUE 12 - STAC (plots, colours), Customization (hardware + software!), PAW (characters, screen printing, flags, externs (sfx)), 6502 Machine Code (loaders, protection), ADLAN (give-to), Z80/68000 Machine Code (converting commands), New Words, etc.

ISSUE 13 - PAW (inventory with objects inside containers listed, characters, objects + containers), Programming Versus Writing, Z80 Machine Code (locations, objects), ADLAN (character sets), Virtual Reality, *Computer Adventures - The Secret Art* review, etc.

ISSUE 14 - CES Show review, Advice For Beginners, PAW (telephone system), ADLAN (starting), Humour In Games, *Splatt!* review, *A Beginners Guide To Adventures* review, Z80 Machine Code (objects, inventory + more), etc.

ISSUE 15 - Mike Gerrard interview, PAW (characters), Adventures (experience, writing styles), ADLAN (variables), etc.

ISSUE 16 - currently out of print! Sorry!

ISSUE 17 - currently out of print! Sorry!

ISSUE 18 - ADLAN (pictures), Tips (Easyscript on the CBM64), Thingy part 2, Creating Puzzles, etc.

ISSUE 19 - currently out of print! Sorry!

ISSUE 20 - STAC (animation, on-screen exits), The Taxman Chronicles part 2, BASIC (strings), Puzzles, *Adventure Link 2* review, PD, Thingy part 4, AMOS (writing a game), PAW (Technical Guide), etc.

ISSUE 21 - GAC (punctuation, descriptions, commands, containers, messages), STAC (wear/remove, objects, combat, colour, WITH command), Polish Computer Magazines review, Thingy part 5, BASIC Text Compression, etc.

ISSUE 22 - Groovy 6510 Adventure Routines For The C64, Contrived Plots, STAC (examine, search, containers, rooms) The Taxman Chronicles part 3, 8-Bits - Dead Or Alive?, AMOS, Thingy part 6, etc.

ISSUE 23 - AMOS, Thingy part 7, Basic Machine Code, loads of letters, The Shredder (new column), Get Real (ideas for adventures) and more!

Regarding the issues out of print at the moment, I am hoping to get these reprinted as soon as I can.

There are also back issues available for the four issues of *Adventure Workshop* that were produced as a sister magazine to this one. They were primarily concerned with 16-bit computers, here's what was in them. (The price is exactly the same as the back issues of *Coder* above!)

A.W. ISSUE 1 - TALESPIN (thoughts), STAC (starting), Programming Versus Writing, AGT review, Z80/68000 Machine Code (converting commands), STAC (thoughts), CES Show review, AMOS (parser, etc), *The Blag* review, *Computer Adventures - The Secret Art* review, etc.

A.W. ISSUE 2 - TALESPIN (starting), STAC (starting, User Routine Access Protocol by Sean Ellis, author of STAC!), AMOS (parser, graphics), Writing Styles, ADVSYS review, Humour In Games, Virtual Reality, Z80/68000 Machine Code (converting commands cont.), etc.

A.W. ISSUE 3 - Magnetic Scrolls exclusive interview!!!, STAC (writing a game, list of faults, exit printing), AMOS (data + editors), TALESPIN (running a game), ST AGT review, Adventure Standards, 68000 Machine Code (addressing modes with table), HATRACK II review, etc.

A.W. ISSUE 4 - Creating Puzzles, Mike Gerrard interview, STAC (object printing), Tips (Star LC-10 printer), Writing Tips, HATRACK II details, GAMESCAPE review, AMOS (useful routines), Adventures (dead?), ST PD, etc.

CODER

Issue 24 December 1995